Blatt 8 - Gruppe: G1-07

Mike Lenz, Jonas Tesfamariam

25. Juni 2023

Aufgabe 1

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.sql.*;
import java.util.ArrayList;
import java.util.Locale;

public class G1_07 {
    public static void main(String args[]) {
        Connection conn = null;
        Configuration conf;

        try {
            conf = Configuration.parse(args);
        } catch( Exception e ) {
            System.out.println("Usage:\n  java GXX_YY path/
                to/file.txt");
            return;
        }
        var results = new ArrayList<Person>();

        try {
            Class.forName("org.postgresql.Driver");
            conn = DriverManager.getConnection(
                    "jdbc:postgresql://" + conf.DB_HOST + "
                        :" + conf.DB_PORT + "/" + conf.
                        DB_NAME, conf.DB_USER,
```

```
                conf.DB_PASSWORD);
        DatabaseMetaData dbmd = conn.getMetaData();
        ResultSet salaryView = dbmd.getTables(null,
            null, "salary" + conf.JAHR, null);

        if (!salaryView.next()) {
            Statement stmt = conn.createStatement();
            stmt.executeUpdate("CREATE VIEW Salary" +
                conf.JAHR
                    + " AS SELECT p.pnr, p.gehalt, CASE
                        WHEN b.betrag = 0 THEN NULL
                        ELSE b.betrag / (SELECT count(*)
                        FROM personal p1 WHERE p.anr =
                        p1.anr) END AS bonus FROM
                        personal p, bonus b WHERE p.anr
                        = b.anr AND b.jahr = "
                    + conf.JAHR);
            stmt.close();
        }

        PreparedStatement pstmt = conn
                .prepareStatement("SELECT s.pnr, p.name
                    , s.gehalt, s.bonus FROM salary" +
                    conf.JAHR + " s, abteilung a,
                    personal p WHERE s.pnr = p.pnr AND p
                    .anr = a.anr AND a.name = ? ORDER BY
                    s.pnr");
        pstmt.setString(1, conf.ABTEILUNG);
        ResultSet res = pstmt.executeQuery();
        while (res.next()) {
            results.add(new Person(res.getInt(1), res.
                getString(2), res.getDouble(3), res.
                getDouble(4)));
        }
        pstmt.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
```

```
        for (var result : results) {
            System.out.println(result);
        }
        try {
            conn.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}

class Configuration {
    public final String DB_NAME;
    public final String DB_HOST;
    public final String DB_PORT;
    public final String DB_USER;
    public final String DB_PASSWORD;

    public final String JAHR;
    public final String ABTEILUNG;

    private Configuration(String name, String host, String
       port, String user, String pw, String year, String
       dept) {
        this.DB_NAME = name;
        this.DB_HOST = host;
        this.DB_PORT = port;
        this.DB_USER = user;
        this.DB_PASSWORD = pw;
        this.JAHR = year;
        this.ABTEILUNG = dept;
    }

    @Override
    public String toString() {
        // Needed for debugging only
        var builder = new StringBuilder();
        builder.append(DB_USER);
        builder.append(':');
        builder.append(DB_PASSWORD);
```

```
        builder.append('@');
        builder.append(DB_HOST);
        builder.append(':');
        builder.append(DB_PORT);
        builder.append('/');
        builder.append(DB_NAME);
        builder.append("?year=");
        builder.append(JAHR);
        builder.append("&abteilung=");
        builder.append(ABTEILUNG);
        return builder.toString();
    }

    public static Configuration parse(String args[]) throws
        IOException {
        if(args.length != 1) {
            throw new IllegalArgumentException();
        }
        var reader = new BufferedReader(new FileReader(args
            [0]));
        String name = reader.readLine().trim();
        String host = reader.readLine().trim();
        String port = reader.readLine().trim();
        String user = reader.readLine().trim();
        String pw = reader.readLine().trim();
        String year = reader.readLine().trim();
        String dept = reader.readLine().trim();
        return new Configuration(name, host, port, user, pw
            , year, dept);
    }
}

class Person {
    private final int pnr;
    private final String name;
    private final double gehalt;
        private final double bonus;

    public Person(int pnr, String name, double gehalt,
       double bonus) {
```

```
        this.pnr = pnr;
        this.name = name;
        this.gehalt = gehalt;
                this.bonus = bonus;
    }

    @Override
    public String toString() {
        var builder = new StringBuilder();
        builder.append(pnr);
        builder.append(", ");
        builder.append(name);
        builder.append(": ");
        builder.append(String.format(Locale.ENGLISH, "%.2f"
            , gehalt));
                builder.append(", ");
                builder.append(String.format(Locale.ENGLISH
                    , "%.2f", bonus));
        return builder.toString();
    }
}
```

Aufgabe 2

a)

i)

Die Funktionale Abhängigkeit $C \rightarrow B$ ist nicht erfüllt, da z.B. bei t1 und t2 C = c6 ist, jedoch B = b2 und B = b8.

ii)

Die Funktionale Abhängigkeit $A \rightarrow C$ ist nicht erfüllt, da z.B. bei t0 und t9 A = a1 ist, jedoch C = c3 und C = c6.

iii)

Die Funktionale Abhängigkeit $CB \rightarrow A$ ist nicht erfüllt, da z.B. bei t8 und t9 B = b3 und C = c6 ist, jedoch A = a4 und A = a1.

iv)

Die Funktionale Abhängigkeit $BCD \rightarrow A$ ist nicht erfüllt, da z.B. bei t8 und t9 B = b3, C = c6 und D = d4 ist, jedoch A = a4 und A = a1.

b)

i)

{C} ist kein Superschlüssel, da bei t1 und t2 C = c6 ist, jedoch B = b2 und B = b8.

ii)

{D} ist kein Superschlüssel, da bei t0 und t3 D = d5 ist, jedoch C = c3 und C = c6.

iii)

{A,D} ist kein Superschlüssel, da bei t6 und t7 A = a2 und D = d2 ist, jedoch B = b6 und B = b7.

iv)

{B,C} ist kein Superschlüssel, da bei t4 und t6 B = b6 und C = c4 ist, jedoch D = d4 und D = d2.

v)

{A,B,C,D} ist ein Superschlüssel, da alle Attribute in der Relation vorkommen und alle Attribute zusammen eindeutig sind. Es ist jedoch kein Kandidatenschlüssel, da {A,B,D} auch ein Superschlüssel ist.

vi)

{A,B,D} ist ein Superschlüssel, da es keine Tupel gibt, bei denen A, B und D gleich sind. Zudem ist es ein Kandidatenschlüssel, denn es gilt:
{A,D} haben wir bereits als kein Superschlüssel identifiziert.
{B,D} ist kein Superschlüssel, da bei t8 und t9 B = b3 und D = d4 ist, jedoch A = a4 und A = a1.
{A,B} ist kein Superschlüssel, da bei t4 und t6 A = a2 und B = b6 ist, jedoch D = d4 und D = d2.
Und folglich sind natürlich {A}, {B} und {D} keine Superschlüssel.

Aufgabe 3

Zu Beweisen: $\alpha \to \beta\gamma \wedge \beta \to \delta \implies \alpha \to \beta\delta\gamma$

1. $\alpha \to \beta\gamma \wedge \beta \to \delta \implies \alpha \to \beta\delta\gamma$ (Gegeben)
2. $\alpha \to \beta\gamma$ (Vereinfachung aus 1.)
3. $\beta \to \delta$ (Vereinfachung aus 1.)
4. $\alpha \to \beta$ (Dekomposition aus 2.)
5. $\alpha \to \delta$ (Transitivität aus 3. und 4.)
6. $\alpha \to \beta\delta\gamma$ (Vereinigungsregel aus 2. und 5.)

Aufgabe 4

a)

Timestamp ist ein Superschlüssel (Und Kandidatenschlüssel da es nur eine Spalte ist), da es keine Tupel gibt, bei denen Timestamp gleich sein kann.

Die restlichen Spalten sind keine Superschlüssel, da selbst SachbearbeiterID, KundeID, Nachricht, Richtung → Timestamp nicht erfüllt ist, denn es ist z.B. möglich für einen Sacharbeiter einem Kunden zwei mal die gleiche Nachricht zu schicken, natürlich zu verschiedenen Zeitpunkten, womit das Wissen über den Sacharbeiter, Kunden, die Nachricht und die Richtung nicht den Timestamp klar definiert.

Also ist die einzige Sinnvolle FD:
Timestamp → SachbearbeiterID, KundeID, Nachricht, Richtung

Andere (Weniger sinnvolle, jedoch trotzdem korrekte) FDs wären z.B.:
SachbearbeiterID, KundeID, Timestamp → Nachricht, Richtung
Nachricht, Richtung, Timestamp → SachbearbeiterID, KundeID
usw.

All diese Regeln lassen sich aus Timestamp → SachbearbeiterID, KundeID, Nachricht, Richtung ableiten.

b)

i)

Die Zerlegung ist verlustlos, da für beide Relationen gilt:
Bei R_1: Timestamp → SachbearbeiterID, KundeID
Bei R_2: Timestamp → Nachricht, Richtung

Zudem ist $R_1 \cap R_2$ = Timestamp wodurch:
Timestamp → SachbearbeiterID, KundeID $\in F_R^+$

Also ist die Zerlegung laut Defininition verlustlos.

ii)

Die Zerlegung ist abhängigkeitserhaltend, da für beide Relationen gilt:
Bei R_1: Timestamp → SachbearbeiterID, KundeID
Bei R_2: Timestamp → Nachricht, Richtung

Laut der Vereinigungsregel können wir bilden:
Timestamp → SachbearbeiterID, KundeID, Nachricht, Richtung

Somit haben wir:
$F_R \equiv (F_{R_1} \cup F_{R_2})$
Also ist die Zerlegung laut Defininition abhängigkeitserhaltend.